(NeXT Tip #51) Our nemesis the PopUpList

Christopher Lane (*lane[at]CAMIS.Stanford.EDU*)
Thu, 12 May 1994 17:35:27 -0700 (PDT)

The PopUpList, one of the later additions to InterfaceBuilder's standard
palette, is also one of the harder to use -- at least in a non trivial way.
Here some hints on how to make PopUpList submit to your will.

If you review how a PopUpList is put together, an InterfaceBuilder PopUpList
involves a chain of objects:

Button => PopUpList => Matrix => your target object

There are other parts & pieces but these are the ones of interest to us. (You
can build a PopUpList piece by piece from code but we'll just look at the
prebuilt IB variety.) The first odd things you learn about PopUpLists are:

1) What you see in InterfaceBuilder is a Button that invokes the PopUpList not
the PopUpList itself. When you set an outlet in InterfaceBuilder, by careful
manipulation of the mouse, you can set it to contain the Button covering for
the PopUpList, the PopUpList itself or one of the items in the PopUpList. You
can tell which you have by the type of object listed in the Connections
browser of the Inspector panel, either 'Button', 'PopUpList' or 'Menu Item'.

Although you can get an outlet to point to any of these, there are application
specific reasons for choosing a particular one to point to. If you want to be
able to set what the currently selected PopUp item is from your code, then you
need an outlet to the Button that covers the PopUpList -- you can get to the
PopUpList from the cover Button (using [button target]) but not vice versa.

If you just want a static PopUpList, you may not need an outlet at all, just
set the targets for the individual PopUp items rather than for the PopUpList.

For the case where you want the PopUpList to have a single target and you
don't plan to manipulate it programmatically, then you should set an outlet
that is the PopUpList itself, because:

2) You cannot set a PopUpList's target from InterfaceBuilder. You need to
include a set* method in your application for the outlet that does:

```
- setMyPopUpList:object
{
    [[(myPopUpList = object) setTarget:self] setAction:@selector(myMethod:)];

    return self;
}
```

But if you set the Button that covers the PopUpList as your outlet, do:

```
    [[[(myPopUpListButton = object) target] setTarget:self] setAction:...];
```

Independent of which outlet object you choose:

3) The message(s) your application receives from a PopUpList come from a
Matrix object, not the PopUpList nor the Button that covers it. Thus, to get
the text string (title) for the item that invoked your method, you should:

```
- myAction:sender
{
    const char *title = [[sender selectedCell] title];
    ...
    return self;
```

```
}
```

NeXT says you can also do [[sender window] selectedItem] to get the text of
the PopUp item but neither they nor I recommend this over the first approach.

Now that we've looked at the basic oddities of using a PopUpList, what are
some of the more subtle ones:

1) How do you programmatically change the currently popped up item? What I've
found that works is:

```
    id myPopUpList = [myPopUpListButton target];

    [myPopUpListButton setTitle:itemName];

    if ([mypopUpList indexOfItem:itemName] == -1)
        [popUpList insertItem:itemName at:0];
```

This changes the contents of the cover button and then inserts the item in the
PopUpList if it isn't already there. You can see why you need a handle on
both the PopUpList and its cover Button for this sort of manipulation.

Caution: Using either approach, the 'selectedItem' won't necessarily reflect
the one you set until the user selects it -- so you'll probably need to use
the cover button's title when you need the currently displayed item instead of
PopUpLists 'selectedItem' method if you manipulate things in this manner.

Also, these currently selected item manipulations don't apply to the alternate
mode of PopUpList which behaves as a pull down menu. (You select the mode
using the less than obviously named method 'changeButtonTitle:' which takes a
boolean argument.) For pull down menu mode, NeXT recommends:

If the title of a pull-down list needs to be changed, both the title
of the trigger Button and the title item of the PopUpList itself need
to be changed. The easiest way to do this is to change the Button's
title, and to remove the title item from the pull-down list with
removeItemAt: (it's kept at position 0).

2) How can I extract the names of the items in a PopUpList?

The way NeXT implemented the PopUpList interface, you can't. (Well you can
but not in any obvious way.) The problem is that although they included the
ability to find out how many items are in the PopUpList, 'count', and how to
get from the index of an item to it's text string, 'indexOfItem:', among other
methods, they left out the method to get the text string of an item given its
index, so you can't simply map over the PopUpList with a 'for' loop.

However there's a fix -- I've cast it below as a category which patches the
PopUpList to add the new method 'itemAt:' as if it were always there:

```
#import <appkit/Matrix.h>
#import <appkit/PopUpList.h>

@implementation PopUpList(PatchMethods)

- (const char *) itemAt:(unsigned int) index
{
    return [[matrix cellAt:index :0] title];
}

@end
```

This backs up a level and searches for the string in the Matrix that's
associated with the PopUpList (you know, the one who's been sending messages
to your application.) With this patch, you can do things like:

```
    unsigned int i, count;

    for (i = 0, count = [myPopUpList count]; i < count; i++) {
        (void) printf("Item %u = %s\n", i, [mypopUpList itemAt:i]);
        }
```

You can find examples of PopUpLists in our online Examples directories, however most of these are limited to trivial usage. You can find some of the strategies discussed above in the source for the Message and Magnify applications which can be found on ~lane/Programming/.

- Christopher

PS: This is being sent to the KSL-NeXT mailing list -- to get off (or onto) this list, send a request to KSL-NeXT-Request[at]CAMIS.Stanford.EDU.